TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

ŘÍDÍCÍ JEDNOTKA PRO SERVOPOHONY DYNAMIXEL AX-12A

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Pavel Dvorský**
Vedoucí práce: Ing. Miroslav Holada, Ph.D.

Liberec 2014

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

THE CONTROL UNIT FOR DYNAMIXEL AX-12A SERVOS

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **Pavel Dvorský**
Supervisor: Ing. Miroslav Holada, Ph.D.

Liberec 2014

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: 2013/2014

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Pavel Dvorský**
Osobní číslo:	**M11000071**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Řídící jednotka pro servopohony Dynamixel AX-12A**
Zadávající katedra:	**Ústav informačních technologií a elektroniky**

Zásady pro vypracování:

1. Seznamte se s problematikou servopohonů stavebnice Bioloid Premium.
2. Navrhněte vlastní řídící jednotku pro serva Dynamixel AX-12A s využitím mikrokontroléru Picaxe.
3. Navrženou řídící jednotku realizujte.
4. Řídící jednotku ověřte v sestavě humanoidního robota a konfrontujte se stávajícím řídícím systémem.

Rozsah grafických prací:	Dle potřeby dokumentace
Rozsah pracovní zprávy:	cca 30 stran
Forma zpracování bakalářské práce:	tištěná/elektronická

Seznam odborné literatury:

[1] Záda, V.: Robotika, matematické aspekty analýzy a řízení. TU v Liberci, Liberec, 2012, ISBN 978-80-7372-882-2

[2] Gook, M.: Hardwarová rozhraní - Průvodce programátora. Computer press, Praha, 2006, ISBN 80-251-1019-2

[3] Corera, A., Fraser, S., McLean, S.: Visual C++ .NET: : A Primer for C++ Developers, Computer Press, ISBN: 978-1861005960, 2003

Vedoucí bakalářské práce:	**Ing. Miroslav Holada, Ph.D.** Ústav informačních technologií a elektroniky
Konzultant bakalářské práce:	**Ing. Zbyněk Mader, Ph.D.** Ústav informačních technologií a elektroniky
Datum zadání bakalářské práce:	**12. září 2013**
Termín odevzdání bakalářské práce:	**16. května 2014**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
vedoucí ústavu

V Liberci dne 12. září 2013

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Rád bych poděkoval vedoucímu práce panu Ing. Miroslavu Holadovi, Ph.D. za poskytnutí materiálů potřebných pro vypracování této práce a jeho rad ohledně směřování projektu.

Abstrakt

Obsahem této práce je úvod do robotiky a seznámení se se stavebnicí Bioloid Premium a servopohony Dynamixel od společnosti ROBOTIS. V práci jsou uvedeny základní informace a rozdíly mezi krokovými motory a servopohony a jaké jsou základní typy komunikací. Dále jsou zde popsány rozdíly a základní technické parametry, které od sebe odlišují jednotlivé řady servopohonů Dynamixel. Hlavní zaměření je pak na typ AX-12A, u kterého jsou uvedeny technické parametry a důkladně pak jsou popsány instrukční a statusové packety, sloužící pro komunikaci mezi servopohony a řídící jednotkou, jak vypadají jednotlivé instrukce a jaké chyby mohou při komunikaci nastat. Práce se dále zaměřuje na popis mikrokontroléru PICAXE a co vedlo k volbě modelu 18M2.

Dalším bodem v této práci je vývoj první verze řídící jednotky, jaké součástky byly zvoleny a proč se od první verze upustilo. Dále práce popisuje chyby, které nastaly při programování druhé verze na nepájivém poli a proč bylo rozhodnuto tuto řídící jednotku sestavit právě na nepájivém poli. Závěrem jsou zde uvedeny ukázky použitých kódu při testování řídící jednotky, jakým způsobem byla zajištěna komunikace mezi řídící jednotkou a servopohony Dynamixel AX-12A, jak byly řešeny problémy v komunikaci a proč je řídící jednotka CM-510 dodávaná se stavebnicí Bioloid Premium nevhodná pro přímé ovládání servopohonů Dynamixel.

Klíčová slova

Servopohon Dynamixel AX-12A, řídící jednotka, PICAXE-18M2, Bioloid Premium, robotika

Abstract

The content of this work is about an introduction to robotics and to familiarize oneself with Bioloid Premium kit and Dynamixel actuators produced by company named ROBOTIS. The paper presents basic information and differences between the stepper motors and servos and what are the basic types of communications. Then there are the differences and basic technical parameters, which distinguish individual series of Dynamixel actuators. The main focus is on the AX-12A type, their technical parameters and then thorough description of the instruction and status packets, which are used for communication between the actuator and the control unit, what does instruction look like and what kind of errors can occur in communication. The work also focuses on the description of PICAXE microcontroller and what led to the choice of model 18M2.

Another point in this work is the development of the first version of the controller, what components were chosen and why was the development of first version abandoned. It also describes the errors that occur when programming the second version constructed on solder field and why it was decided to construct a controller unit just to solder field. Finally, there are examples of the codes used in the testing of the control unit, how the communication between the control unit and servo Dynamixel AX-12A was ensured, how problems in communication were resolved and why the control unit CM-510 supplied by the kit Bioloid Premium is unsuitable for direct control of Dynamixel actuators.

Keywords

Dynamixel AX-12A servo, control unit, PICAXE-18M2, Bioloid Premium, robotics

Obsah

Seznam obrázků

Seznam tabulek

Seznam vzorců

Seznam zdrojových kódů

1. Úvod

V dnešní době jsou roboti součástí každodenního života. Z knihy R.U.R., jejímž autorem je Karel Čapek, se pod pojmem robot představuje stroj podobný člověku. V dnešní době se sice ve výzkumných centrech provádí výzkumy na zkonstruování robota co nejvíce podobného člověku jak po stránce hardwarové, tak i po stránce softwarové, tedy umělá inteligence fungující nezávisle na lidském faktoru.

Robot ale není jenom stroj podobný člověku. Může mít různé podoby. Například dnes je lze najít v mobilních telefonech, automobilech, ve většině moderních domácích spotřebičů, automatech na kávu a na mnoha dalších místech.

Pojem robot obecně označuje takový kus technologie, který samostatně vykoná předem stanovený čin bez nutnosti zásahu lidského faktoru. Používají se pro zpracování vstupních dat a na základě jejich výsledků provést určitou akci, pro ovládání motorů či třeba kontrolu periferních zařízení.

Jelikož jsou roboti nezbytnou součástí dnešní doby a budou i v budoucnosti, je potřeba se je naučit konstruovat a ovládat. To vede k cíli této práce. Na pracovišti školitele je stavebnice Bioloid Premium od společnosti ROBOTIS. Ze zkušeností z minulých let bylo zjištěno, že řídící jednotka CM-510 dodávaná s touto stavebnicí není vhodná pro využívání mimo tuto stavebnici. Kvůli způsobu programování této řídící jednotky je velmi složité ovládat motory v reálném čase. To je důvod vzniku této práce, která si dává za cíl vytvoření nové řídící jednotky pro motory výše zmiňované stavebnice.

V tomto dokumentu je shrnuta robotika a pojem robot. Dále je zde popsána stavebnice Bioloid Premium od společnosti ROBOTIS, její základní charakteristické údaje a rozdíly oproti dalším verzím společně se zkušenostmi z předcházejících let s touto stavebnicí a softwarem určeným pro její programování. Součástí této stavebnice jsou servopohony Dynamixel, na které je zaměřena tato práce. Je zde tedy obsažen krátký úvod k pohonům, rozdíly mezi servopohony a krokovými motory a stručný popis sériové a paralelní komunikace. Dále je zde popsána základní charakteristika včetně vlastností servopohonů Dynamixel a stručný popis jednotlivých řad servopohonů Dynamixel nabízené společností ROBOTIS se zaměřením na typ AX-12A, u kterého jsou

uvedeny technické parametry a způsob komunikace. Kromě servopohonů je zde také popsán mikrokontrolér PICAXE-18M2 a proč byl zvolen.

Třetí kapitola popisuje vývoj návrhu a realizace řídící jednotky, vznik jednotlivých verzí, proč byl vývoj první verze ukončen před zhotovením a jaké výhody přinesl jiný přístup u následující verze. Dále popisuje problémy, které se v průběhu vývoje objevily, a také jak byly řešeny.

2. Robotika

Robotiku lze v dnešní době považovat za vědu zabývající se vývojem robotických zařízení z hledisek konstrukčního (hardwarového) a programovacího (softwarového). Robotické zařízení (neboli robot) je takové zařízení, které samostatně vykonává určitou činnost.

Roboti mají v dnešní době mnoho podob. Mohou to být humanoidní roboti předvádějící sekvenci pohybů jakožto prezentaci svých schopností, ale i obyčejné průmyslové rameno u pásu, jehož úkolem je přemístit určitý objekt z pásu například do přepravky.

Roboti se dají rozdělit do dvou kategorií a to na roboty měnící svou polohu a roboty neměnící svou polohu. První kategorii představují roboti s koly, pásy nebo nohama. Mezi nejznámější představitele patří humanoidní roboti nebo robotické vozítko Mars rover. Hlavním představitelem druhé kategorie je průmyslové rameno. Jedná se o nejrozšířenějšího robota, který byl vynalezen.

V této práci bylo využito stavebnice Bioloid Premium od společnosti ROBOTIS. Z této stavebnice byl sestaven humanoidní robot typu C dle návodu poskytnutého výrobcem, který měl následně posloužit pro ověřování funkčnosti nové řídící jednotky.

2.1. Stavebnice Bioloid Premium

Obrázek 1: Humanoidní robot typu C (zdroj: BIO_PRM_Humanoid_ASM.pdf na přiloženém CD)

Stavebnice Bioloid Premium je z hlediska vývoje třetí stavebnicí ze čtyř řady Bioloid od korejské společnosti ROBOTIS. Její výhodou oproti předcházejícím stavebnicím, nebo též verzím, je větší počet motorů a součástek, které nabízí mnohem větší sortiment robotů, které lze z této stavebnice sestavit (kromě 3 typů humanoidního robota například ještě škorpión, pes a mnohem více), a také záměna senzoru AX-S1 za infračervený senzor, Distance Measurement Sensor (zkráceně DMS) pro měření vzdálenosti od 10 cm do 80 cm a v neposlední řadě gyroskop, který funguje na principu akcelerometru. Poslední stavebnicí řady je Bioloid GP. Tato verze je rozšířena o další součástky a hlavně nový typ servopohonů Dynamixel AX-18A.

Hlavním účelem stavebnice Bioloid je seznámit širokou veřejnost s robotikou. Využívá se ve školách a institucích zabývajících se robotikou, aby přiblížila funkční principy robotů a jejich programování včetně možnosti pochopení používaných algoritmů s možností jejich vylepšení či vytvoření zcela nových. Tato stavebnice ale sklidila velký úspěch a tak se mimo výzkumné zaměření stala terčem pro fanoušky robotiky a často se využívá v různých soutěžích.

2.1.1. Praktické zkušenosti

Humanoidní robot typu C se skládá z 16 servopohonů Dynamixel AX-12A rozdělených do 4 samostatných větví pro každou končetinu. Konstrukce kyčelních kloubů sice omezuje tento typ robota v některých pohybech, jako je například otočení na místě, ale poskytuje mu větší stabilitu než nabízejí verze typu A a B. Zásadní problém činí řídící jednotka CM-510 s baterií a několika motory v horní části robota. Tím je poměrně velká váha uložena v horní části robota, což posouvá do horní části i samotné těžiště. To znamená sníženou stabilitu robota a náchylnost na nerovnost povrchu při pohybu.

Samotné programování řídící jednotky CM-510 dodávané výrobcem se skládá ze dvou částí, kde se každá programuje v jiném programu. Oba tyto programy a další jsou součástí jednoho balíčku RoboPlus. Mezi ty nejdůležitější programy patří software Task, Motion, Manager a Dynamixel Wizard.

První část programování je psaní programu ve stylu jazyka C. Pro toto programování se používá software Task a v tomto programu jsou obsaženy všechny

proměnné, podmínky a funkce. Jelikož je tato stavebnice určena pro vzdělávací účely na školách i pro amatérské kutily, snažili se výrobci upravit tento software tak, aby byl pokud možno jednoduchý a snadno pochopitelný, uživatelsky přívětivý, nedaly se udělat zbytečné chyby a byla dodržena správná syntaxe. Všechny tyto požadavky nakonec vedly k mnoha omezením. Hlavním omezením je nemožnost ručně napsat kus kódu. Vše je řešeno formou takového dotazníku. Uživatel si nejprve otevře menu s příkazy, kde si zvolí konkrétní příkaz, který potřebuje (například podmínka If). Pokud přesně nezná anglický název zvolené funkce (například funkce Load pro deklaraci a inicializaci proměnné), může se v tomto menu ztratit. Po zvolení požadovaného příkazu software sám vygeneruje syntaxi a na místech proměnných zanechá otazníky. Zdrojový kód 1 uvádí příklad podmínky IF

```
IF(? == ? then)
{

}
```

Zdrojový kód 1: Vygenerovaný zdrojový kód podmínky IF

Uživatel následně musí vybrat parametr pro otevření nového menu s možnostmi, které lze na toto místo vložit. V případě funkce IF nabízí první otazník možnosti od vlastních proměnných po proměnné a konstanty získané z periferních zařízení (dálkové ovládání, stisknutá tlačítka, časovač, hodnoty na servopohonu a další). Druhým parametrem je porovnání hodnot. Zde uživatel může zvolit možnosti jako je rovno (==), je různé od (!=), je větší (>) či menší (<) a další. Třetím parametrem je druhý otazník, který stejně jako první otazník nabízí konstanty a proměnné. Čtvrtým parametrem je then, který lze změnit na logický AND (&&) nebo OR (||). Parametr then označuje konec podmínky IF, kdežto AND a OR přidají znovu první 3 parametry pro další podmínku. Tento styl programování má výhodu, že uživatelům je jasně stanovená cesta, ze které nelze sejít. Syntaxe programu tedy bude vždy správná (chybu lze udělat pouze při mazání řádků) a je zajištěno, že nemůže dojít k překlepům. Nevýhodou je ale omezení ručního zásahu do kódu. Dialogový výběr všech proměnných a jejich konkrétních hodnot zabere podstatně více času než v případě ručně psaného kódu a pro zkušenější uživatele je tato uživatelská jednoduchost na obtíž.

Druhá část programování je tvorba sekvencí, které bude řídící jednotka vykonávat. Software Motion obsahuje celkem 255 stránek a každá tato stránka se skládá až ze sedmi kroků a dalších parametrů, jako je například počet opakování či rychlost provedení. Dále ještě kromě svého ID a jména obsahují ID následující stránky a ID stránky, kterou má tato sekvence skončit. Každý krok v sobě obsahuje cílovou polohu všech servopohonů, do které se mají otočit, pauzu před provedením kroku a čas, jak dlouho se daný krok bude vykonávat. Tento software umožňuje ručně nastavit cílové polohy jednotlivých servopohonů nebo, v případě připojeného robota, získat a uložit aktuální polohu servopohonů. V záložce Pose Utility je také grafické zobrazení robota (v nabídce je výběr ze 17 různých druhů robotů), které simuluje zadanou polohu jednotlivých servopohonů a také umožňuje ručně hýbat s jednotlivými servopohony. Hlavní předností v tomto programování je možnost při tvorbě sekvencí umístit robota do požadované polohy a aktuální hodnoty uložit do nového kroku, což značně zjednoduší samotnou tvorbu kroků.

Software Manager je upravená verze softwaru Dynamixel Wizard. Díky tomuto programu je možné sledovat aktuální hodnoty všech připojených periferií, ať už se jedná o servopohony nebo senzory, a také umožňuje změnit některé hodnoty. Například u servopohonů lze nastavit cílovou polohu. Tento software je závislý na připojení řídící jednotky CM-510 a všech periferií k ní. Jelikož řídící jednotka má pevně stanovenou rychlost komunikace 1 Mbps, neumožňuje tedy tento software změnit například přenosovou rychlost (baud rate), zpoždění pro odeslání statusového packetu po přijetí instrukčního packetu (return delay time) a další. Ze zkušeností je funkce tohoto programu spíše pro pochopení jednotlivých hodnot nebo také funkcí připojených periferií a jejich testování než pro samotné ovládání servopohonů, protože lze sledovat nebo ovládat vždy pouze jednu periferii.

Software Dynamixel Wizard není závislý na řídící jednotce, jako tomu je u softwaru Manager, a tedy umožňuje změnit všechny hodnoty v řídící tabulce. Výjimky tvoří pouze ty hodnoty, které jsou nastavené pouze pro čtení. To jsou například aktuální poloha a rychlost otáčení, vnitřní teplota, modelové číslo, verze firmwaru a další. Tento software oproti softwaru Manager neumožňuje práci s jinými periferiemi, než jsou servopohony Dynamixel, a kvůli možnosti změnit všechny hodnoty v řídící jednotce,

které jsou nastaveny do režimu čtení/zápis (RW), je označován jako program pro pokročilé. Důvodem je například změna přenosové rychlosti, kvůli které by daný servopohon nebyl v zapojení s řídící jednotkou CM-510 funkční. Jelikož řídící jednotce navržené v této práci nelze nastavit přenosová rychlost na 1 Mbps a to ani s 3% rozdílem, je tento software nezbytně nutný pro přenastavení potřebných hodnot servopohonů, aby se zajistila funkční komunikace.

Do řídící jednotky CM-510 jsou tedy nahrány 2 samostatné programy. První program ze softwaru Task (.tsk) obsahuje funkce, podmínky a proměnné, se kterými se dále pracuje. Druhý program je vytvořen v softwaru Motion (.mtn) a obsahuje stránky s polohami servopohonů, které vytvoří sekvence pohybů. První program pomocí speciálních proměnných zavolá konkrétní stránku z druhého programu a robot tak vykoná určitý pohyb. Takto je řešena například chůze robota nebo předvádění určitých pohybů, jako je zatleskání nebo postavení se po pádu zpět na nohy. Problémem ale je nemožnost změny programu za chodu. Zahájený program musí být ukončen, řídící jednotka musí být přepnuta do režimu programování a teprve potom lze do řídící jednotky nahrát nový program. Poté se musí opět ručně přepnout do režimu prezentace, aby mohl být program spuštěn. Dalším problémem je nemožnost ovládat servopohony v reálném čase. Řídící jednotka umožňuje volání předem definovaných pohybů ze souboru .mtn, ale při pokusu ovládat servopohony ručně pomocí dálkového ovládání docházelo k chybám a servopohony se například místo otáčení po dobu stisknutého tlačítka otáčely neustále. Tento software nebyl koncipován pro takové ovládání a způsob programování v softwaru Task i značně ztěžuje ladění vzniklých chyb. Z těch zkušeností také vyplynulo, že tuto řídící jednotku nelze dále rozšiřovat o nové moduly.

2.2. Servopohony a krokové motory

Jednou ze stěžejních funkcí robotů je jejich pohyb. Ať už mají změnit svou fyzickou polohu nebo se pouze natočit, je na jejich pohyb kladen velký důraz. Tento pohyb je zajištěn pomocí motorů a servopohonů.

Jedním z rozdílů mezi krokovými motory a servopohony je zpětná vazba. Servopohony oproti krokovým motorům poskytují zpětnou vazbu, díky které lze zjišťovat aktuální informace o konkrétním servopohonu v reálném čase. Je možné

tak zjistit například aktuální teplotu motoru a pokud by přesáhla předem stanovenou hodnotu, může se z bezpečnostních důvodů motor odstavit z provozu, díky čemuž lze předejít jeho poškození. Tato zpětná vazba přináší řadu výhod díky informacím, které lze z motoru získat, ale přináší také nevýhody. Zpětnou vazbu zprostředkovává nějaká elektronika, která ovlivní velikost a cenu servopohonu, takže ne vždy jsou servopohony správnou volbou. Dále mezi řídící jednotkou a servopohonem musí probíhat oboustranná komunikace, která může přinést řadu problémů.

Dalším rozdílem je samotné ovládání zemi krokovými motory a servopohony. Krokové motory jsou ovládány připojením napětí na jednotlivé cívky v motoru, čímž se zajistí jednotlivé kroky. Servopohony jsou ovládány pomocí packetů s pevně danou strukturou. Tento packet je rámec dat tvořící jednu jednotku v sériové komunikaci a obsahuje informace, kde servopohonu nastaví požadovanou cílovou polohu.

2.2.1. Paralelní komunikace

Hlavní výhodou paralelní komunikace je posílání více bitů najednou. U 8bitového paralelního portu to znamená, že za dobu odeslání jednoho bitu se odešle bitů 8. Oproti sériové komunikaci to tedy znamená 8krát větší rychlost, bude-li zachována stejná přenosová rychlost[2].

Odesílání více bitů zároveň ale přináší své nevýhody. Může se stát, že data přijdou s různým zpožděním a tedy nebudou dávat smysl – dojde k jejich poškození. To přineslo omezení pro délku kabelu na maximální vzdálenost 30 stop (v přepočtu zhruba 9,1 m). Paralelní komunikace se dále dělí na 2 protokoly. Prvotní standardní paralelní port (SPP), který umožňoval pouze jednosměrnou komunikaci, a následně vylepšený paralelní port (EPP, Enhanced Parallel Port), který už umožňoval oboustrannou komunikaci a zvýšil tak možnou rychlost přenosu dat z původních 150 kb/s na 2 Mb/s.

2.2.2. Sériová komunikace

Oproti paralelní komunikaci se u sériové posílají data postupně za sebou. Její výhodou je snadná synchronizace dat, která se dělí na synchronní a asynchronní přenos. Synchronní přenos je zajišťován společným hodinovým signálem pro vysílač i přijímač a využívá se převážně pro přenos větších objemů dat. U asynchronního přenosu mají

vysílač i přijímač vlastní hodiny a k synchronizaci dochází před každým odeslaným bytem. Prvnímu bitu odesílaných dat předchází start-bit a za posledním bitem dat následuje stop-bit. V některých případech se k datům přidává i paritní bit, který slouží

ke kontrole znaku (dat), zdali nedošlo k chybě.

Obousměrná sériová komunikace se dělí na 2 části, full duplex a half duplex. Full duplexní komunikace umožňuje vysílat i přijímat data zároveň. Z toho vyplývá největší výhoda full duplexní komunikace, kterou je rychlost. Nevýhodu ale tvoří nutnost samostatného kanálu pro každý směr komunikace. Half duplexní komunikace probíhá po jednom kanálu, což tedy znamená, že data se mohou buď vysílat nebo přijímat, ale nikdy současně. V obvodu tedy musí být umístěn řídící prvek, který rozhoduje o aktuálním směru komunikace. Výhodou je využití jediného kanálu a oproti full duplexní komunikaci větší kompatibilita. Nevýhodu tvoří nižší rychlost komunikace.

2.3. Dynamixel

Dynamixel jsou všestranné servopohony vyvinuty korejskou společností ROBOTIS, hojně využívané na univerzitách i mezi nadšenci po celém světě. Na svou poměrně malou velikost nabízí vysokou přesnost, robustnost a hlavně vysoký točivý moment[1]. Nabízí také zpětnou vazbu, kdy uživatel může sledovat aktuální stav servopohonu, jako je například aktuální poloha, rychlost otáčení, teplota motoru a další. Podle typu lze napájet tyto servopohony stejnosměrným napětím od 6 V do 18 V, nejčastěji v rozmezí 9 ~ 12 V. Právě toto poměrně vysoké napětí pro modelářské motory a servopohony dává servopohonům Dynamixel výhodu. Toto poměrně vysoké napětí se běžně používá a je tedy jednodušší jej sehnat. Komunikace je zajištěna asynchronní sériovou komunikací o délce jednoho rámce 8 bitů, 1 stop-bit a žádný paritní bit. Dále také jednotlivé typy používají half duplexní TTL nebo RS-485 signál.

Servopohony Dynamixel obsahují řídící tabulku (control table), která slouží k jejich ovládání. Uchovávají se v ní veškeré informace o servopohonu, jako je například jeho ID, verze aktuálně nahraného firmwaru, aktuální poloha, cílová poloha, teplota a další. Skládá se ze dvou různých pamětí. První je nevolatilní EEPROM, kde jsou uloženy informace, které musí být zachovány i po odpojení motoru od napájení (ID, verze firmwaru, rychlost komunikace a další). Druhou pamětí je volatilní paměť typu RAM.

V ní jsou uloženy informace, které nejsou nezbytně důležité, a po připojení servopohonu k napájení se jim nastaví defaultní hodnoty (aktuální poloha, teplota, cílová poloha a další). Samotné ovládání motoru je řešeno změnou hodnot v této řídící tabulce instrukčním packetem.

2.3.1. Dostupné řady

Společnost ROBOTIS aktuálně nabízí 6 řad servopohonů Dynamixel, a to AX, DX, EX, MX, RX a XL-320. Řada XL-320 nabízí v tuto chvíli pouze 1 stejnojmenný servopohon napájen 6 ~ 8,4 V. Jedná se o malý 3pinový servopohon pracující na TTL logice, jehož točivý moment činí 0,39 N·m při napětí 7,4 V. Tento servopohon lze přepnout do režimu kola nebo kloubu. Pracovní úhel v kloubovém režimu činí 0° ~ 300°, zatímco režim kola má otáčivý úhel nekonečný. Rychlost komunikace se uvádí 7 343 bps ~ 1 Mbps, ale je možné nastavit pouze hodnoty 9 600 bps, 57 600 bps, 115 200 bps nebo 1 Mbps.

Řada RX nabízí 3 typy servopohonů, které se liší váhou, rozměry, napájecím napětím (9 ~ 12 V a 12 ~ 18,5 V), rychlostí otáčení naprázdno a točivým momentem. Komunikace probíhá pomocí RS-485 logiky. Tento servopohon je možné nastavit do režimu kola nebo kloubu, kdy pracovní úhel je 0° ~ 300°. Typ RX-24F (napájecí napětí do 12 V) má při maximálním napětí točivý moment a rychlost otáčení 2,6 N·m a 126 rpm. Pro typy RX-28 a RX-64 (napájecí napětí do 18,5 V) jsou to hodnoty 3,7 N·m (5,3 N·m pro RX-64) a 85 rpm (64 rpm pro RX-64). Rychlost komunikace lze nastavit v rozsahu 7 343 bps ~ 1 Mbps. Starší typ RX-10 se již nevyrábí ani neprodává.

Řada MX, stejně jako předcházející řada RX, nabízí 4 motory, které se liší v rozměrech, hmotnosti, převodovém poměru, točivém momentu a otáčení naprázdno. Stejně jako předchozí typy je zde možnost nastavit servopohon do režimu kloubu nebo kola. Napájecí napětí je pro všechny motory v rozmezí 10 ~ 14,8 V a kromě typu MX-12W, který používá pouze logiku TTL, mají všechny typy servopohonů verze s logikou TTL nebo RS-485. Na základě logiky se jedná o 3pinový (TTL) nebo 4pinový (RS-485) servopohon. Jedinečnost řady MX je v provozním úhlu v kloubovém režimu, konkrétně 0° ~ 360°, a rychlosti komunikace, která činí 8 kbps ~ 4,5 Mbps. Při napětí 12 V je rychlost otáčení naprázdno servopohonu MX-12W 470 rpm. Točivý moment a rychlost otáčení pro zbylé typy jsou 2,5 N·m a 55 rpm (MX-28T/R), 6 N·m a 63 rpm

(MX-64T/R), 8,4 N·m a 45 rpm (MX-106T/R). Písmena T a R značí logiku TTL nebo RS-485.

Řada EX nabízí pouze jeden typ motoru EX-106+ napájen 12 ~ 18,5 V. Rychlost komunikace lze nastavit v rozsahu 7 343 bps ~ 1 Mbps. Jeho přední vlastností je velmi vysoký točivý moment 10,9 N·m a rychlost 91 rpm při maximálním napětí. Jedná se o větší a poměrně těžký servopohon pracující na RS-485 logice. Tento motor lze nastavit do režimu kola nebo kloubu. Oproti ostatním řadám má nejmenší pracovní úhel v rozsahu 0° ~ 251°, což činí tuto řadu z pohledu přesnosti natočení nejpřesnější.

Řada DX nabízí typ DX-117 napájen 12 ~ 18,5 V. Rychlost komunikace lze nastavit v rozsahu 7 343 bps ~ 1 Mbps. Pro komunikaci využívá RS-485 logiku a hlavním rozdílem oproti ostatním řadám je absence módu pro přepnutí mezi režimy kolo a kloub. Je zde pouze kloubový režim s pracovním uhlem 0° ~ 300°. Při maximálním napětí je točivý moment 3,7 N·m a rychlost otáčení 85 rpm. Starší typy DX-113 a DX-116 se již nevyrábí ani neprodávají.

Řada AX nabízí 3 typy servopohonů, které jsou napájeny napětím 9 ~ 12 V. Rychlost komunikace je nastavitelná v rozsahu 7 343 bps ~ 1 Mbps na TTL logice. Jednotlivé typy se liší hlavně rychlostí otáčení a točivým momentem, konkrétně 54 rpm a 0,2 N·m (AX-12W), 59 rpm a 1,5 N·m (AX-12A) a 97 rpm a 1,8 N·m (AX-18A). Stejně jako u předchozích řad (kromě DX) je zde možnost nastavit servopohon do režimu kola nebo kloubu s pracovním úhlem 0° ~ 300°.

2.3.2. Dynamixel AX-12A

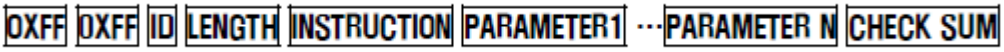
Servopohonům Dynamixel AX-12A předcházely verze AX-12 a AX-12+. Mezi jednotlivými verzemi je rozdíl převážně v designu a převodu. Technické parametry těchto servopohonů jsou popsány v Tabulka 1. Řídící tabulka obsahuje celkem 43 adres, 18 pro paměť typu EEPROM a zbylých 25 pro paměť typu RAM. Kompletní strukturu řídící tabulky lze najít na přiloženém CD na straně 12 souboru AX-12.pdf.

Tabulka 1: Technické parametry servopohonu Dynamixel AX-12A

Technické parametry	Specifické hodnoty
Hmotnost [g]	54,6
Rozměry [mm]	32 × 50 × 40
Rozlišení [°]	0,29
Převodový poměr	254/1
Točivý moment při 12 V, 1,5 A [N·m]	1,5
Počet otáček naprázdno při 12 V [rpm]	59
Pracovní úhel v režimu kloubu [°]	0 ~ 300
Pracovní úhel v režimu kola [°]	Nekonečné otáčení
Provozní teplota [°C]	-5 ~ +70
Napájecí napětí [V]	+9 ~ +12
Doporučené napájecí napětí [V]	+11,1
Řídící signál	Digitální packet
Protokol	Asynchronní sériová half duplex komunikace
Datový packet	8 bitů, 1 stop-bit, žádný paritní bit
Řídící logika	TTL
Počet adres (ID)	254 (0 ~ 253)
Rychlost komunikace [bps]	7 343 ~ 1 M

2.3.2.1. Instrukční packet

Instrukční packet je takový soubor dat, který obsahuje informace určené pro jeden nebo v případě vysílání jako broadcast (ID 0xFE) pro všechny servopohony. Tato data obsahují typ instrukce (akce, kterou má servopohon vykonat), ID servopohonu, pro který jsou data určena, doplňující parametry a kontrolu správnosti instrukce. Samotný instrukční packet se skládá ze 6 nebo více bytů. Jeho strukturu lze vidět na Obrázek 2.



Obrázek 2: Struktura instrukčního packetu (zdroj: http://support.robotis.com/en/product/dynamixel/communication/dxl_packet.htm)

První dva byty (0xFF, 0xFF) označují začátek packetu a pro servopohony to znamená, že přichází instrukce. V tuto chvíli všechny servopohony připojené na sběrnici přijímají data.

Třetí byte ID určuje příjemce, tedy konkrétní servopohon, pro který je daná instrukce určena. Servopohony porovnají toto příchozí ID s ID uloženém ve své paměti typu EEPROM (na adrese 0x03) a pokud se neshodují, tak ukončí příjem dat a čekají na další hlavičku začínající byty 0xFF a 0xFF. Výjimkou je hodnota 254 (0xFE), která značí broadcast, tedy vysílání pro všechny servopohony připojené na sběrnici. Pokud je ID

rovno hodnotě 254, všechny servopohony vykonají příchozí instrukci a nevrátí statusový packet.

Čtvrtý byte LENGTH označuje délku příchozí instrukce, tedy konkrétně kolik následujících bytů má servopohon přečíst. Jedná se o součet všech dodatečných parametrů u instrukce a hodnoty 2 (viz vzorec 1), která zastupuje byte INSTRUCTION a byte CHECK SUM.

$$LENGTH = PARAMETER\ 1 + PARAMETER\ 2 + \ldots + PARAMETER\ N + 2$$

(Vzorec 1)

Byte INSTRUCTION obsahuje kód konkrétní instrukce, kterou servopohon vykoná. Seznam instrukcí, jejich kód a počet parametrů je uveden v Tabulka 2. Podle typu instrukce mohou následovat dodatečné parametry (na Obrázek 2 označené jako PARAMETER) v následujících N bytech, kdy N označuje celkový počet parametrů. V těchto parametrech se uvádí například adresa v paměti řídící tabulky, kolik bytů se má číst nebo jaké hodnoty se mají zapsat.

Tabulka 2: Seznam instrukcí

Hodnota	Instrukce	Počet parametrů
0x01 (1)	PING	0
0x02 (2)	READ_DATA	2
0x03 (3)	WRITE_DATA	2 nebo více
0x04 (4)	REG WRITE	2 nebo více
0x05 (5)	ACTION	0
0x06 (6)	RESET	0
0x83 (131)	SYNC WRITE	4 nebo více

Poslední byte CHECK SUM zastupuje kontrolu, zdali je instrukční packet v pořádku. Jedná se o negovaný součet všech předchozích bytů v instrukci kromě hlavičky (viz vzorec 2). Pokud je součet před negací větší než velikost jednoho bytu (větší než 255), vezme se pouze spodní byte.

$$\sim CHECK\ SUM = ID + LENGTH + INSTRUCTION + PARAMETER\ 1 + \ldots + PARAMETER\ N$$

(Vzorec 2)

2.3.2.2. Popis instrukcí

Instrukce PING nevykonává sama o sobě žádnou akci a používá se pouze ve dvou případech. Prvním případem je oznámení řídící jednotky servopohonu, že je připravena přijmout status packet. Druhým případem je kontrola, zdali je požadovaný servopohon připojen ke sběrnici. Pokud ano, obdrží zpět statusový packet.

Instrukce READ_DATA slouží k využití zpětné vazby poskytované servopohony Dynamixel a tedy k získání hodnot z řídící tabulky uložené v servopohonu. Tato instrukce má právě dva parametry. První parametr označuje adresu v řídící tabulce (například hodnota 0x03 pro zjištění ID servopohonu) a druhý parametr označuje počet bytů, které se budou číst (v případě čtení ID by druhý parametr měl hodnotu 0x01).

Pomocí instrukce WRITE_DATA se de facto ovládají servopohony. Tato instrukce zapisuje hodnoty do řídící tabulky servopohonu a tak určuje jeho akci. Obsahuje vždy dva nebo více parametrů. První parametr udává adresu v paměti řídící tabulky, na kterou se začnou zapisovat data. Ve druhém parametru je obsažena konkrétní hodnota, která se zapíše na adresu z prvního parametru. Každý následující parametr udává konkrétní hodnotu, která se zapíše na následující adresu. Pokud by byla například hodnota prvního parametru 0x03 (ID servopohonu), uloží se hodnota třetího parametru na adresu 0x04 (nastavení baud rate), hodnota čtvrtého parametru se uloží na adresu 0x05 (return delay time – čas, za jak dlouho servopohon vrátí statusový packet od přijetí posledního bytu instrukce) atd.

Instrukce REG WRITE funguje stejně jako předchozí instrukce WRITE_DATA s tím rozdílem, že instrukce se neprovede okamžitě. Místo toho se uloží dočasné paměti a na adrese 0x2C v řídící tabulce (zapsané v seznamu, registered) se nastaví hodnota 1. Poté se čeká na instrukci typu ACTION, při které se na adrese 0x2C nastaví hodnota 0 a uložená instrukce se provede. Tato instrukce je vhodná pro zajištění lepšího časování při ovládání servopohonů, kdy se pomocí broadcast vysílání pošle instrukce ACTION všem servopohonům připojeným ke sběrnici.

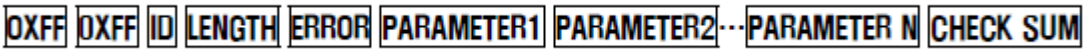
Instrukce ACTION, jak již bylo zmíněno v předchozím odstavci, slouží pro spuštění instrukce uložené v dočasné paměti a výhodou jejího využívání eliminace časových rozdílů mezi jednotlivými instrukcemi v případě ovládání více servopohonů současně.

Instrukce RESET slouží čistě pro nastavení servopohonu do továrního nastavení. To se zajistí přepsáním hodnot v řídící tabulce. U používání tohoto příkazu je nutné si dávat pozor, neboť může dojít ke změně přenosové rychlosti, což může mít za následek nefunkční komunikaci s řídící jednotkou, která využívá jiné přenosové rychlosti, než je tovární nastavení servopohonu. Také může dojít ke změně ID servopohonu a tedy instrukční packety používané doposud pro resetovaný servopohon nebudou fungovat.

Instrukce SYNC WRITE se využívá k ovládání více servopohonů současně odesláním jednoho instrukčního packetu. Tato instrukce musí mít vždy minimálně čtyři parametry, vysílá se jako broadcast pro všechny servopohony a v případě vysílání pro více servopohonů se skládá z více instrukcí. Prvním parametrem je adresa v paměti řídící jednotky, na kterou se začnou data zapisovat. Druhý parametr určuje počet bytů, které náleží jedné instrukci. Důležité je podotknout, že se do tohoto čísla nezapočítává ID, ale pouze samotná data pro zápis, a v případě posílání dat pro více servopohonů platí tato délka pro každou instrukci. Třetí parametr určuje ID servopohonu, pro který jsou následující data určena a čtvrtý parametr jsou samotná data. V případě dvou nebo více servopohonů naváže za posledním parametrem předcházející instrukce nový parametr označen jako třetí, tedy parametr ID, na který navazují data pro další servopohon. Pro lepší pochopení je uveden jednoduchý příklad v příloze A.

2.3.2.3. Statusový packet

Statusový packet je soubor dat vyslaný servopohonem, který obsahuje informace určené pro řídící jednotku. Skládá se z hlavičky, ID, své délky, chyby a kontrolního bytu. Jeho strukturu lze vidět na Obrázek 3.



Obrázek 3: Struktura statusového packetu (zdroj: http://support.robotis.com/en/product/dynamixel/communication/dxl_packet.htm)

Statusový packet je velice podobný instrukčnímu packetu. Začíná dvěma 0xFF byty jako hlavičkou oznamující začátek nového packetu. Dalším bytem je ID servopohonu, který statusový packet odeslal. Po ID následuje byte LENGTH určující počet následovaných bytů do konce packetu. Výpočet je shodný jako u instrukčního packetu (viz vzorec 1).

Rozdílem oproti instrukčnímu packetu je následující byte ERROR. Tento byte obsahuje informace o chybě, která nastala během provádění operace. V případě toho bytu je nezbytné jej převést do binární soustavy a porovnávat hodnoty jednotlivých bitů (hodnota 1 znázorňuje chybu, hodnota 0 že k chybě nedošlo), neboť každý bit znamená jiný druh chyby. V Tabulka 3 je bližší popis jednotlivých chyb.

Tabulka 3: Seznam chyb a jejich bitová reprezentace

Bit	Název	Popis
Bit 7	0	Tento bit je vždy nulový
Bit 6	Chyba instrukce (Instruction Error)	Tato chyba se nastaví v případě, kdy servopohon obdrží neznámou instrukci nebo instrukci typu ACTION, které nepředcházela instrukce REG WRITE
Bit 5	Chyba přetížení (Overload Error)	Tato chyba se nastaví v případě, když při momentálním zatížení nelze ovládat servopohon dle zadaného točivého momentu řídící tabulkou
Bit 4	Chyba kontrolního bytu (Checksum Error)	Tato chyba se nastaví v případě, kdy byte checksum příchozího instrukčního packetu je chybný
Bit 3	Chyba rozsahu (Range Error)	Tato chyba se nastaví v případě, když příkaz z instrukčního packetu přesáhne povolený rozsah
Bit 2	Chyba přehřátí (Overheating Error)	Tato chyba se nastaví v případě, když aktuální vnitřní teplota servopohonu přesáhne hranici nastavenou v řídící tabulce
Bit 1	Chyba maximálního úhlu (Angle Limit Error)	Tato chyba se nastaví v případě, pokud je cílová pozice mimo rozsah stanovený úhlovými limity (CW Angle Limit a CCW Angle Limit) uloženými v řídící tabulce
Bit 0	Chyba vstupního napětí (Input Voltage Error)	Tato chyba se nastaví v případě, když vstupní napětí je mimo rozsah pracovního napětí

Následujících N bytů zastupují parametry, tedy konkrétní data, o která si řídící jednotka zažádala instrukčním packetem. Počet N se lze jednoduše zjistit z hodnoty bytu LENGTH. Výpočet je uveden ve vzorci 3. Poslední byte, stejně jako u instrukčního packetu, tvoří byte CHECK SUM pro kontrolu správnosti vysílaných dat. Výpočet tohoto bytu je shodný jako u instrukčního packetu (viz vzorec 2).

$$N = LENGTH - 2$$

(Vzorec 3)

2.4. Mikrokontrolér

Mikrokontrolér je malý čip, často označován jako integrovaný obvod. Používá se jako řídící jednotka určitého zapojení a nebo jako součást jiného zapojení, kde vykonává

nějakou specifickou funkci. Jedná se o kompaktní řešení rozsáhlého zapojení, které kromě vstupně/výstupní pinů poskytuje například převodník z analogového vstupu na digitální data (AD převodník) a obráceně (DA převodník), možnost měnit frekvenci oscilátoru, pulzně-šířkovou modulaci (PWM), čítače a časovače a další.

Mikrokontroléry se nejčastěji používají ve vestavných (anglicky embedded) systémech, často napájených baterií. Díky tomu jsou na ně kladeny nároky jako nízká spotřeba, malý rozměr či vysoký rozsah pracovní teploty. Dalším požadavkem je i nízká cena, která je ovlivněna velikostí a funkcemi nabízenými mikrokontrolérem.

Hlavní výhodou mikrokontroléru je jeho univerzálnost a kompaktnost. Mnoho obvodů, jako je například AD převodník, je již integrovaných v tomto čipu a tedy není potřeba je přidávat do zapojení externě. Tím se sníží výrobní náklady a velikost potřebného prostoru pro takové zapojení.

Příliš mnoho funkcí ale znamená volbu mezi větším pouzdrem nebo spojením více funkcí na jeden pin, což znamená možnost použít pouze jednu funkci na pin. První nevýhodou je tedy volba funkce, která se na daném pinu bude používat. Další nevýhodu přináší malý rozměr, tedy nízký počet pinů, kdy problémy nastanou při snaze o rozšíření tohoto zapojení. Obě tyto nevýhody jsou však vyřešeny širokou nabídkou různých druhů mikrokontrolérů na trhu, takže je možné si vybrat takový mikrokontrolér, který co nejvíce vyhovuje požadavkům a potřebám.

2.4.1. PICAXE-18M2

Mikrokontrolér PICAXE byl původně navržen pro studijní účely, aby se studenti středních a vyšších škol a univerzit seznámili s elektronikou a také programováním jednoduchých zapojení. Dnes se však hojně používá i mezi amatéry a profesionály díky svým nízkým nákladům a jednoduchému programování. Seznam všech instrukcí je k dispozici na oficiálních stránkách.

Mikrokontrolér PICAXE vychází z mikrokontroléru PIC od společnosti Microchip a jedná se v podstatě o mikrokontrolér PIC s již přednahraným kódem, který umožňuje programování mikrokontroléru PICAXE přímo v zapojení. Pro programování se využívá 3vodičový kabel (Rx, Tx, GND) a samotné programování probíhá po sériové komunikaci.

Oproti mikrokontrolérům PIC zde není potřeba použít speciální programátor, což je jednou z předností mikrokontrolérů PICAXE.

Další předností je jednoduchý programovací jazyk BASIC s volně dostupným programovacím softwarem, který je univerzální pro všechny mikrokontroléry PICAXE. Výhodou tohoto programovacího jazyku oproti jazykům C a assembler, které se běžně používají pro programování mikrokontrolérů, je jeho jednoduchost pro pochopení a také snadné ladění (debug).

Pro tuto práci byl vybrán model PICAXE-18M2. Důvodem je velikost čipu. Tento model má celkem 18 pinů, které dále nabízí 16 vstupně/výstupních pinů s 10 analogovými kanály. Tento počet dostačuje pro navrhnuté zapojení v této práci a také nechává několik pinů jako rezervních pro případné další rozšíření. Volba mezi modelem 18 a 18M2 odpadla z důvodu nedostupnosti starého modelu 18. Navíc model 18M2 nabízí vyšší přenosové rychlosti včetně několika dalších programových vylepšení. Například příkaz `serin` obsahuje navíc parametr `timeout`, který určuje čas, jak dlouho bude program čekat na příchozí data, než bude pokračovat. Seznam všech instrukcí lze najít na stránkách výrobce[4].

3. Realizace řídící jednotky

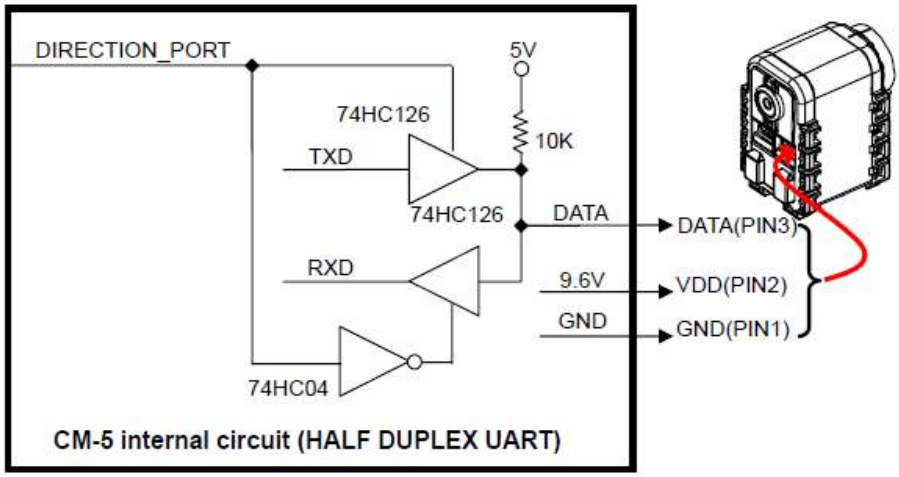
V této kapitole jsou popsány jednotlivé verze návrhů při vytváření schémat, desek plošných spojů a kódů pro ovládání servopohonů. Nejprve je uveden postup tvorby schémat a návrhů pro desky plošných spojů. Druhá podkapitola popisuje použitý kód a jeho vývoj pro programování servopohonů včetně získaných hodnot. V poslední kapitole jsou uvedeny použité programy.

3.1. První verze návrhu řídící jednotky

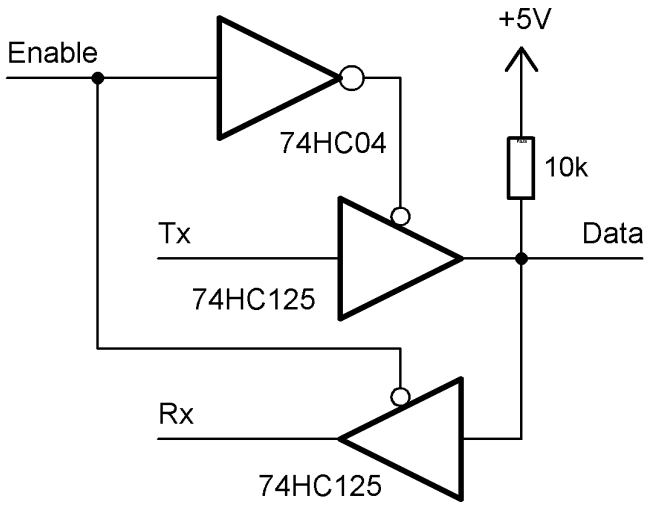
Jelikož komunikace mezi řídící jednotkou a servoponohy Dynamixel probíhá po sériové komunikaci s protokolem half duplex, je nezbytné sestavit obvod, který převede obousměrný signál typu UART z řídící jednotky na jednosměrný signál typu half duplex. Schéma pro tento obvod vycházelo z doporučeného zapojení uváděného výrobcem. Kvůli špatné dostupnosti integrovaného obvodu 74HC126, který v doporučeném zapojení uváděl výrobce, byl použit lépe dostupný integrovaný obvod 74HC125. Oba tyto integrované obvody fungují na principu brány pro řízení signálu a každý z nich obsahuje celkem 4 takové brány. Každá brána se skládá ze vstupního pinu pro data, výstupního pinu pro data a pinu enable, který rozhoduje, zdali obvod uzavře (příchozí data se ztratí) a nebo obvod otevře a data propustí. Jediný rozdíl mezi těmito součástkami je negovaný pin enable u integrovaného obvodu 74HC125. To znamená, že integrovaný obvod 74HC126 se otevře signálem o hodnotě 1, zatímco integrovaný obvod 74HC125 se otevře signálem o hodnotě 0. Aby se zachovala původní myšlenka pro odesílání dat při logické hodnotě enable 1 a příjem dat při logické hodnotě 0, muselo dojít ještě k jedné změně oproti doporučenému zapojení. Invertor (integrovaný obvod 74HC04), který byl původně u pinu enable hradla pro příjem dat (RXD na Obrázek 4Obrázek 4), bylo nezbytné přesunout k hradlu pro odesílání dat (Tx na Obrázek 5). U odesílání dat tak vznikla dvojitá negace.

Účel první desky byl vytvořit jednoduchý převodník pro half duplex komunikaci, ke které bude možné připojit mikrokontrolér picaxe a servopohon Dynamixel. Po vytvoření schématu a návrhu desky plošných spojů se nakonec od této verze upustilo (viz kapitola 3.1) a začalo se pracovat na verzi, kde již bude zabudován i samotný

mikrokontrolér. Schéma doporučeného zapojení a první verze lze vidět na Obrázek 4 a Obrázek 5.



Obrázek 4: Doporučené zapojení pro převodník UART - half duplex [zdroj: http://support.robotis.com/en/product/dynamixel/dxl_ax_main.htm]



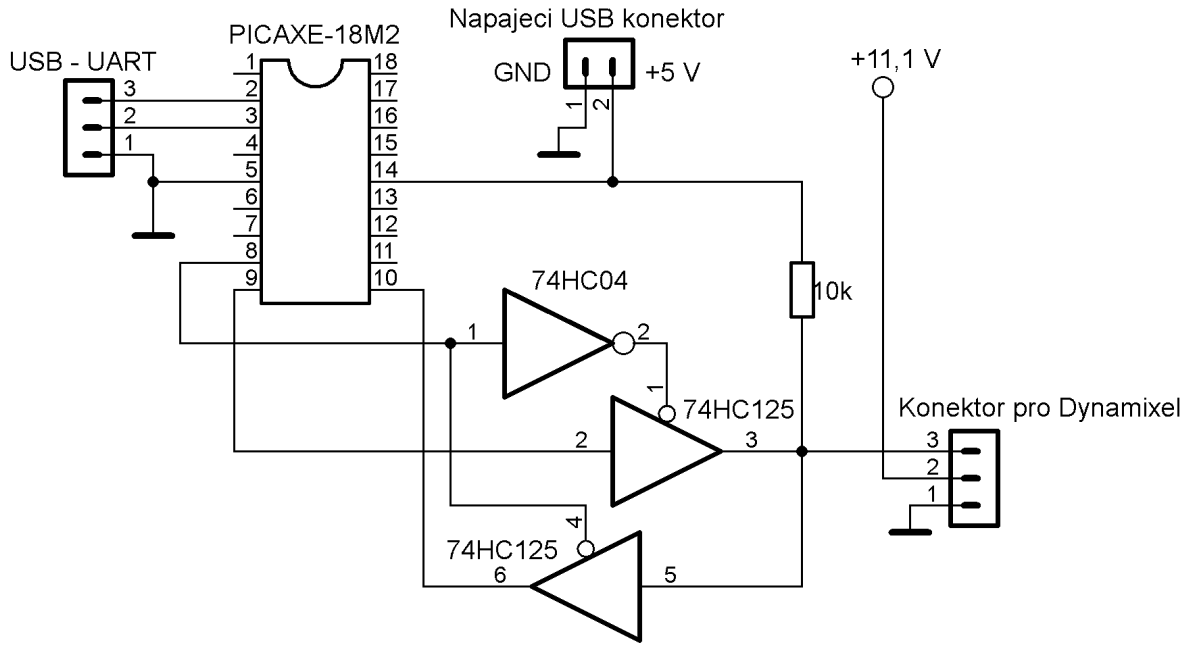
Obrázek 5: Schéma upraveného zapojení pro první desku

3.2. Verze s využitím nepájivého pole

Jelikož první verze se ukázala vhodná pouze jako připojitelný modul, bylo rozhodnuto o návrhu nového zapojení, které bude vhodné pro testování a co nejvíce se bude podobat finální verzi. Toto zapojení by již mělo obsahovat samotný picaxe, komunikační převodník z předchozí verze, dva konektory pro připojení dvou samostatných sériových větví servopohonů Dynamixel, konektor pro připojení baterie pro napájení servopohonů Dynamixel, konektor pro napájení obvodu z USB a konektor typu COM pro programování mikrokontroléru PICAXE 18M2.

3.2.1. Fyzická část

Po vytvoření schématu bylo rozhodnuto o použití nepájivého pole místo vyrobení desky plošných spojů. Toto rozhodnutí přineslo několik výhod a změn. První výhodou bylo zkrácení času potřebného pro zhotovení zapojení. Odpadla potřeba návrhu a samotné konstrukce desky plošných spojů. Další výhodou byla možnost ladění chyb přímo na místě, bez nutnosti fyzického zásahu do desky. Například pokud by u zapojení došlo k chybě, mohla se takováto chyba opravit pouhým přepojením drátku. V případě desky plošných spojů by tato oprava byla značně náročnější. Třetí a hlavní výhodou byla snadná možnost přizpůsobení a úpravy při testování. To znamená například možnost připojit LED diodu pro signalizaci logické hodnoty 1 nebo 0 v zapojení či připojení osciloskopu pro sledování signálu na daném úseku v obvodu. Výhodou byla možnost testování zapojení modulárně po jednotlivých částech.



Obrázek 6: Schéma zapojení na nepájivém poli

3.2.2. Programová část a ladění

Když bylo vše zapojeno, bylo potřeba nahrát do mikrokontroléru program, který bude ovládat motory. Zde se objevil první problém. Program se skládal z povolení enable pinu a jednoduchého instrukčního packetu pro servopohon, který měl rozsvítit LED diodu zabudovanou v servopohonu Dynamixel. Jak se dalo pro první pokus očekávat, program nefungoval.

První možností, v čem mohla být chyba, se jevila špatná správa sběrnice bus, po které probíhá komunikace. Z pohledu řídící jednotky byla pevně nastavena do režimu vysílání. To znamenalo, že řídící jednotka vyslala data, ale už žádná nebyla schopná přijmout. Data, která motor vyslal zpět v podobě statusového packetu, byla ztracena. Tato myšlenka se opravila přepnutím řídící jednotky do přijímacího režimu ihned po odeslání dat a přijmutí šesti bajtů (délka statusového packetu jako odpověď v případě, že servopohon přijal a vykonal instrukci bez chyb), ale nebyla řešením pro vzniklý problém, neboť řídící jednotka nepřijala žádná data.

Druhou možností bylo špatné pořadí posílaných dat v packetu, tedy použití malého endianu. Po vyzkoušení prohození vysílaných bytů a prostudování příkazu serin s instrukčním packetem pro Dynamixel vyšlo najevo, že v této možnosti chyba není.

Z výsledku druhé možnosti vyplynula možnost třetí, a to konkrétně špatná rychlost komunikace mezi řídící jednotkou a servopohonem Dynamixel. Použitý mikrokontrolér PICAXE 18M2 pracuje v základu na frekvenci 4 MHz s maximální přenosovou rychlostí 4 800 bps, ale je možné jej interně přetaktovat až na 32 MHz s maximální přenosovou rychlostí 38 400 bps. Servopohony Dynamixel AX-12A mají z výroby nastavenou rychlost komunikace na 1 Mbps. Aby mohla komunikace fungovat, musí být všechny prvky (servopohony i řídící jednotka) nastaveny na stejnou přenosovou rychlost. Z toho faktu vyplývá první nedostatek této navrhované řídící jednotky. U mikrokontroléru PICAXE 18M2 nelze nastavit dostatečně vysokou přenosovou rychlost (a to ani přidáním externího krystalu), aby se daly servopohony Dynamixel AX-12A jakkoli ovládat a tedy u nich ani nelze snížit přenosovou rychlost. Je nutné využít jiného postupu pro změnu přenosové rychlosti servopohonů Dynamixel. Jako první možnost se jevilo využití řídící jednotky CM-510 dodané se stavebnicí Bioloid Premium, ale zde se objevil problém, že v jednotce nebylo umožněno změnit přenosovou rychlost. Po prozkoumání této problematiky na internetu byla nalezena pouze jedna možnost, jak tuto přenosovou rychlost přenastavit. V elektronickém manuálu[3] na stránkách společnosti ROBOTIS, v sekci pro stavebnici DARwIn-OP, je položka pro výměnu servopohonů Dynamixel řady MX. Protože jsou všechny řady servopohonů Dynamixel v jádru velice podobné, funguje tento návod i pro řadu AX. Pro změnu přenosové rychlosti tedy bylo zapotřebí připojit servopohon Dynamixel k řídící jednotce CM-510 a modulu USB2Dynamixel v TTL

režimu, který se používá pro programování řídící jednotky CM-510. Poté se v programu RoboPlus (software dodávaný společností ROBOTIS pro programování řídící jednotky CM-510) spustil program Dynamixel Wizard, ve kterém byla možnost změnit přenosovou rychlost daného servopohonu na požadovanou hodnotu.

Hodnota přenosové rychlosti byla zvolena na 9 600 bps. V případě servopohonu se tato hodnota nastavuje dle vzorce 4, kde Data znamenají bytovou hodnotu 0 ~ 254. Pro Data o hodnotě 207 vychází přenosová rychlost (Speed) 9 615,4 bps, která se nejvíce blíží hodnotě 9 600. Maximální tolerance rozdílu přenosových rychlostí jsou 3 %, což je v tomto případě dodrženo. V případě mikrokontroléru jej pro zajištění přenosové rychlosti 9 600 bps stačilo přetaktovat z původních 4 MHz na 8 MHz a u příkazů serout a serin zvolit jako parametr baud rate požadovanou hodnotu T9600_8 nebo N9600_8 podle požadované polarity.

$$Speed\left[bps\right] = \frac{2000000}{Data + 1}$$

(Vzorec 4)

Po nastavení stejné přenosové rychlosti již mikrokontrolér přijímal data, ale místo očekávaných hodnot se jednalo o náhodná čísla. Pro ladění tohoto problému se nastavila v programu nekonečná smyčka, ve které se po jedné vteřině vysílal instrukční packet pro rozsvícení LED diody na servopohonu, a pomocí příkazu debug se sledovaly jednotlivé byty příchozího statusového packetu. Tento problém nakonec byl v používání invertované polarity (parametr baud rate N9600_8) a byl vyřešen změnou na skutečnou polaritu (parametr baud rate T9600_8).

V Tabulka 4 jsou uvedeny hodnoty jednotlivých bytů, které byly přijaty řídící jednotkou po vyslání instrukčního packetu servopohonu pro rozsvícení LED diody zabudované v servopohonu. Každý sloupec zastupuje hodnoty statusového packetu získané po odeslání jedné instrukce. Hodnoty 1 až 3 byly přijaty při přenosové rychlosti 1 Mbps nastavené na straně servopohonu a 9,6 kbps s invertovanou polaritou na straně řídící jednotky . Přenosové rychlosti byly rozdílné a tedy přijatá data byla poškozena. Hodnoty 4 a 5 byly přijaty při stejných přenosových rychlostech jako hodnoty 1 až 3 s rozdílem otočení polarity na skutečnou na straně řídící jednotky. Hodnota 6 byla získána při nastavení přenosové rychlosti na 9,6 kbps na straně řídící jednotky

i servopohonu (v případě servopohonu konkrétně na 9 615,4 bps) s invertovanou polaritou. Zde je vidět, že přijatá data se už začínala blížit očekávaným hodnotám s tím rozdílem, že byla invertovaná. Data z hodnoty 7 byla obdržena při stejném nastavení jako v případě hodnoty 6 s rozdílem obrácené polarity, která tedy byla nastavena na skutečnou. Z těchto hodnot lze vidět start packetu (B1 a B2), ID 14 servopohonu, od kterého přišel statusový packet (B3), zbývající počet 2 bytů do konce packetu (B4), instrukční packet přijatý servopohonem byl v pořádku a nedošlo k žádné chybě (B5) a byte CHECK SUM (B6) má stejnou hodnotu, kterou získáme výpočtem ze vzorce 2. Jelikož se obdržený CHECK SUM i vypočítaný CHECK SUM rovnají, přijatý statusový packet byl v pořádku.

Tabulka 4: Hodnoty přijatého statusového packetu

Byte	Hodnota 1	Hodnota 2	Hodnota 3	Hodnota 4	Hodnota 5	Hodnota 6	Hodnota 7
B1	23	253	145	3	249	1	255
B2	255	251	253	255	9	0	255
B3	251	255	255	255	255	188	14
B4	255	255	253	0	0	223	2
B5	255	255	253	0	255	255	0
B6	255	255	254	255	0	195	239

3.2.3. Testovací kódy

Když byly všechny potřebné kroky pro zajištění komunikace mezi řídící jednotkou a servopohonem podniknuty, bylo na čase začít vytvářet kódy, které budou ovládat servopohony. Zdrojový kód 2 ukazuje část první verze programu pro testování komunikace. Konkrétně návěští, které rozsvítilo LED diodu a přijalo statusový packet. Úkolem tohoto programu bylo v nekonečné smyčce rozsvítit LED diodu na servopohonu ID 12 po dobu 1 vteřiny a poté ji zhasnout.

```
l_on:
      high b.2
      serout b.3,T9600_8,(0xff)
      serout b.3,T9600_8,(0xff)
      serout b.3,T9600_8,(0x0c)
      serout b.3,T9600_8,(0x04)
      serout b.3,T9600_8,(0x03)
      serout b.3,T9600_8,(0x19)
      serout b.3,T9600_8,(0x01)
      serout b.3,T9600_8,(0xd2)
      low b.2
      serin b.4,T9600_8,b1,b2,b3,b4,b5,b6
      debug
      pause 2000
```

Zdrojový kód 2: Ukázka kódu programu blik_led_v1.bas

Tato první verze programu nefungovala jak by měla. Problém se ukázal v příkazu `serin`. Program čeká na přijetí 6 bytů znázorňujících statusový packet. Ten však byl odeslán dříve, než si jej řídící jednotka všimla a tak je celý program zaseknut, kdy řídící jednotka očekává data od servopohonu a servopohon očekává nový instrukční packet od řídící jednotky. Takto vzniklá situace se anglicky nazývá deadlock. Tento problém byl vyřešen v druhé verzi (viz zdrojový kód 3), kde se příkazu `serin` přidaly parametry `timeout` a `adress`.

```
l_on:
      high PEN
      serout POUT,BAUD,(0xff)
      serout POUT,BAUD,(0xff)
      serout POUT,BAUD,(0x0c)
      serout POUT,BAUD,(0x04)
      serout POUT,BAUD,(0x03)
      serout POUT,BAUD,(0x19)
      serout POUT,BAUD,(0x01)
      serout POUT,BAUD,(0xd2)
      low PEN
      serin [200,l_on],PIN,BAUD,b1,b2,b3,b4,b5,b6
      debug
      pause 2000
```

Zdrojový kód 3: Ukázka kódu programu blik_led_v2.bas

Oproti předchozí verzi zde byly jednotlivé hodnoty nahrazeny proměnnými, čímž se zajistila větší přehlednost kódu a možnost jednoduché úpravy kódu v případě volby jiného pinu. Hlavním rozdílem jsou ale přidané parametry u příkazu `serin`. Tyto příkazy

říkají, že program bude čekat na přijatá data 100 ms a pokud data nepřijdou, provede se příkaz znovu. Frekvence mikrokontroléru je nastavena na dvojnásobek základní hodnoty. To znamená, že se všechny instrukce provádí dvakrát rychleji a tedy čekání po dobu 200 ms se provede za poloviční dobu, tudíž 100 ms.

Zdrojový kód 4 ukazuje část kódu, který otáčí servopohonem ID 14 v nekonečné smyčce. Smyčka se skládá ze 4 kroků. První krok rozsvítí LED diodu jakožto signalizaci provozu servopohonu a přivede napětí na jeho cívky. Ve druhém kroku se servopohon otočí do své výchozí polohy, kterou je střed jeho pracovního úhlu, rychlostí 57 rpm. Třetí krok se skrádá z dalších 3 mezikroků, kdy se s každým mezikrokem servopohon otočí o přibližně 30° proti směru hodinových ručiček. Ve čtvrtém kroku se zhasne LED dioda a odpojí se napětí z cívek servopohonu.

```
;rotate 30° ccw
l_rotate_30:
      high PEN
      serout POUT,BAUD,(0xff,0xff,0x0e,0x07)
      serout POUT,BAUD,(0x03,0x1e,xL,xH,0x00,0x02,checksum)
      low PEN
      serin [2000,l_rotate_30],PIN,BAUD,b1,b2,b3,b4,b5,b6
      return

;rotate 3 times
l_rotate_ccw:
      for w10 = 612 to 812 step 100
            let xL = b20
            let xH = b21
            let checksum = 56 + xL + xH
            let checksum = 255 - checksum
            gosub l_rotate_30
            debug
            pause 1000
      next w10
```

Zdrojový kód 4: Ukázka kódu programu rotate.bas

V této ukázce je vidět použití proměnné checksum pro výpočet bytu CHECK SUM, který se s každým otočením o 30° změní. Z funkce l_rotate_30 lze vyčíst, že součet hodnot v packetu od ID až po poslední parametr bez samotných hodnot cílové polohy je konstanta, která je použita při výpočtu bytu checksum (56). K této hodnotě se přičtou samotné hodnoty cílové polohy (proměnné xL a xH znázorňující spodní a horní byte) a následně se hodnota invertuje jednoduchým odečtením od maximální hodnoty jednoho bytu, kterou je 255.

V programu rotate_3.bas bylo použito stejného kódu jako v programu rotate.bas, ale poprvé se řídící jednotka testovala se 3 zapojenými servopohony najednou. Kód byl tedy rozšířen o další funkce, které spouštěly 2 nově připojené servopohony. Během testování tohoto kódu se vyskytla chyba, kdy se program zasekl u servopohonu ID 12. Problém se nakonec ukázal ve špatně nastaveném zpoždění pro odesílání statusového packetu. Defaultně byla tato doba nastavena na 0 μs a řídící jednotka se nestihla přepnout do režimu pro příjem dat.

3.2.4. Ověření řídící jednotky v sestavě

V tomto bodě je malé odbočení od zadání. Původně měla být řídící jednotka otestována na humanoidním robotovi na pracovišti vedoucího projektu, ale až

v průběhu práce se zjistilo, že servopohony musí být přenastaveny, což znamenalo volbu mezi řídící jednotkou navrženou v této práci nebo řídící jednotkou CM-510 dodávanou se stavebnicí Bioloid Premium. Humanoidní robot se využívá na různých akcích jako jsou například dny otevřených dveří nebo veletrhy Educa a Gaudeamus, kde slouží pro prezentaci Technické univerzity. Z toho důvodu bylo rozhodnuto odbočit od zadání a řídící jednotku otestovat na zapojení uchopovače sestaveného v průběhu minulých let.

Program gripper.bas ovládá 6 servopohonů rozdělených do 2 ramen. Pro otestování byly zvoleny 2 různé instrukce, pro každé rameno jiná. U prvního ramena se posílají instrukční packety postupně jednotlivým servopohonům pomocí instrukce WRITE_DATA, které je ihned vykonávají. To má za následek zpoždění mezi pohyby jednotlivých servopohonů. U druhého ramena je využita instrukce REG WRITE, díky které se nejdříve zapíší instrukce do jednotlivých servopohonů a poté se instrukcí ACTION vykonají všechny najednou. V programu je nahraná smyčka, která přiblíží a následně oddálí jednotlivá ramena.

4. Závěr

Tato práce se zabývá úvodem do robotiky a seznámením se se stavebnicí Bioloid Premium od společnosti ROBOTIS. Popisuje základní rozdíly mezi krokovými motory a servopohony, jaké existují typy komunikací a důkladněji se zaobírá vlastnostmi a rozdíly mezi jednotlivými řadami servopohonů Dynamixel nabízených společností ROBOTIS, konkrétně pak typem AX-12A. Dále jsou zde důkladně popsány instrukční a statusové packety, které zprostředkovávají komunikaci mezi servopohonem a řídící jednotkou a zároveň slouží pro ovládání servopohonů, jaká je jejich struktura a co jaký byte znamená. Kromě servopohonů je tato práce zaměřena i na mikrokontrolér PICAXE, jaké jsou jeho výhody oproti ostatním mikrokontrolérům a jaké požadavky vedly k volbě modelu 18M2. Dále je zde popsán vývoj první verze řídící jednotky, kdy se jednalo spíše o přídavný modul než samotnou řídící jednotku, a její návrh a konstrukční řešení. Vývoj druhé řídící jednotky je již spojen s jejím programováním a jsou zde popsány chyby, které se v průběhu programování vyskytly, a jak byly řešeny. Konkrétně se jednalo o špatnou správu směru komunikace po sběrnici, špatně nastavenou přenosovou rychlost komunikace a příliš malé zpoždění pro odeslání statusového packetu. Špatná správa sběrnice znamená, že řídící jednotka byla nastavena do režimu vysílání i když nic nevysílala, kdežto pro správný chod komunikace po sériové sběrnici musí být všechny připojené prvky v přijímacím režimu.

Přenosová rychlost byla dlouho překážkou. Všechny pokusy změnit tuto přenosovou rychlost na servopohonu pomocí řídící jednotky selhaly. Problémem bylo, že mikrokontrolér PICAXE-18M2 lze vnitřně přetaktovat z výchozích 4 MHz až na 32 MHz, ale nelze přidat externí krystal pro nastavení vyššího kmitočtu. To znamená, že při maximálním kmitočtu dosahuje mikrokontrolér maximální přenosové rychlosti 38 400 bps. Aby mohla komunikace probíhat v pořádku, musí být rozdíl přenosových rychlostí mezi prvky maximálně 3 %. Servopohony Dynamixel mají od výroby nastavenou přenosovou rychlost 1 Mbps, po které komunikují s řídící jednotkou CM-510. Rozdíl mezi těmito dvěma přenosovými rychlostmi činí přibližně 96 %. Problém byl nakonec vyřešen použitím modulu USB2Dynamixel, s jehož pomocí lze v programu Dynamixel Wizard z balíčku RoboPlus přenastavit tuto hodnotu na požadovaných 9 600 bps. Stejným způsobem bylo přenastaveno i zpoždění pro odesílání

statusového packetu na 500 µs, neboť defaultních 0 µs bylo pro mikrokontrolér příliš rychlé, kvůli čemuž se v čekací smyčce pro příjem dat dostal do tzv. deadlocku. Tato práce také uvádí ukázky použitých kódů s vysvětlením jejich principu a jaké problémy řešily a popisuje zkušenosti s programováním řídící jednotky CM-510.

Řídící jednotka navržena v této práci nebyla nakonec otestována dle zadání na humanoidním robotovi. Místo toho byla otestována na sestavě uchopovače sestaveného studenty z předešlých let. Důvodem pro odbočení od tématu byl fakt, že zmiňovaný humaniodní robot je využíván na různých akcích pro reprezentaci Technické univerzity a ústavu Informačních technologií a elektroniky. V případě otestování na humanoidním robotovi by bylo potřeba zvolit, která řídící jednotka by se používala, neboť pro každou řídící jednotku je nezbytné jiné nastavení servopohonů Dynamixel. Aby se předešlo zbytečným problémům, bylo rozhodnuto otestovat toto zapojení na sestavě uchopovače, který poskytuje velice podobné prostředí pro otestování. V zapojení této sestavy se navržená řídící jednotka ukázala funkční.

Řídící jednotka je v tuto chvíli volná pro další možné rozšíření, jako je například přidání možnosti pro připojení dalších periferií (různé senzory či komunikační moduly) nebo vytvoření nového softwaru pro její řízení a programování.

Použitá literatura

Záda, V.: Robotika, matematické aspekty analýzy a řízení. TU v Liberci, Liberec, 2012, ISBN 978-80-7372-882-2

Gook, M.: Hardwarová rozhraní - Průvodce programátora. Computer press, Praha, 2006, ISBN 80-251-1019-2

Corera, A., Fraser, S., McLean, S.: Visual C++ .NET: : A Primer for C++ Developers, Computer Press, ISBN: 978-1861005960, 2003

PLÍVA, Zdeněk, Jindra DRÁBKOVÁ, Jan KOPRNICKÝ a Leoš PETRŽÍLKA. *Metodika zpracování bakalářských a diplomových prací*. Technická univerzita v Liberci, 2009 [cit. 2014-04-13]. ISBN 978-80-7494-049-1.

Seznam zdrojů

[1] Advantage of Dynamixel actuators?. In: *Trossen Robotics Community* [online]. 2013 [cit. 2014-05-03]. Dostupné z: http://forums.trossenrobotics.com/showthread.php?5988-Advantage-of-Dynamixel-actuators

[2] Parallel Communication Overview. *B&B Electronics Mfg* [online]. 2014 [cit. 2014-05-08]. Dostupné z: http://www.bb-elec.com/Learning-Center/All-White-Papers/Serial/Parallel-Communication-Overview.aspx

[3] Replacing Dynamixel(s). ROBOTIS. *ROBOTIS e-Manual* [online]. 2010 [cit. 2014-05-10]. Dostupné z: http://support.robotis.com/en/product/darwin-op/self-maintenance/replacing_dynamixel(s).htm

[4] *PICAXE* [online]. 2014 [cit. 2014-05-12]. Dostupné z: http://www.picaxe.com/BASIC-Commands/

Přílohy

CD obsahující elektronickou verzi práce ve formátu .pdf a .doc, kopii originálního zadání v .pdf, manuál BIO_PRM_Humanoid_ASM.pdf a zdrojové kódy programů blik_led_v1.bas, blik_led_v2.bas, rotate.bas, rotate_3.bas a gripper.bas.

A Příklad instrukčního packetu

Příklad: Nastavení přenosové rychlosti servopohonů ID 1, 2 a 3 na 9 615,4 bps.

Instrukční packet: 0xFF 0xFF 0xFE 0x0A 0x83 0x04 0x01 0x01 0xCF 0x02 0xCF 0x03 0xCF 0xFC

Vysvětlení bytů:

Hlavička: 0xFF 0xFF

ID: 0xFE (broadcast)

Délka instrukce: 0x0A (počet následujících bytů)

Instrukce SYNC WRITE: 0x83

Zápis na adresu: 0x04 (baud rate)

Délka jedné instrukce: 0x01

První instrukce:

ID servopohonu: 0x01

Hodnota pro zápis: 0xCF

Druhá instrukce:

ID servopohonu: 0x02

Hodnota pro zápis: 0xCF

Třetí instrukce:

ID servopohonu: 0x03

Hodnota pro zápis: 0xCF

CHECK SUM: 0xFC

Výpočet CHECK SUM:

Součet hodnot: $0x\left[FE+0A+83+04+01+01+CF+02+CF+03+CF=403\right]$

Jelikož hodnota 0x0403 je větší než 255, bere se pouze spodní byte – 0x03.

CHECH SUM = ~ 0x03 = 0xFF – 0x03 = 0xFC